Software Development for the Grape Personal Space Weather Station v2

Cuong Nguyen KC3UAX

Dr. Nathaniel Frissell W2NAF, John Gibbons N8OBJ, Bill Blackwell AB1XB,

Bill Engelke AB4EJ, HamSCI Community, Dr. Mark Fenner

THE UNIVERSITY OF SCRANTON
A JESUIT UNIVERSITY

cuong.nguyen@scranton.edu

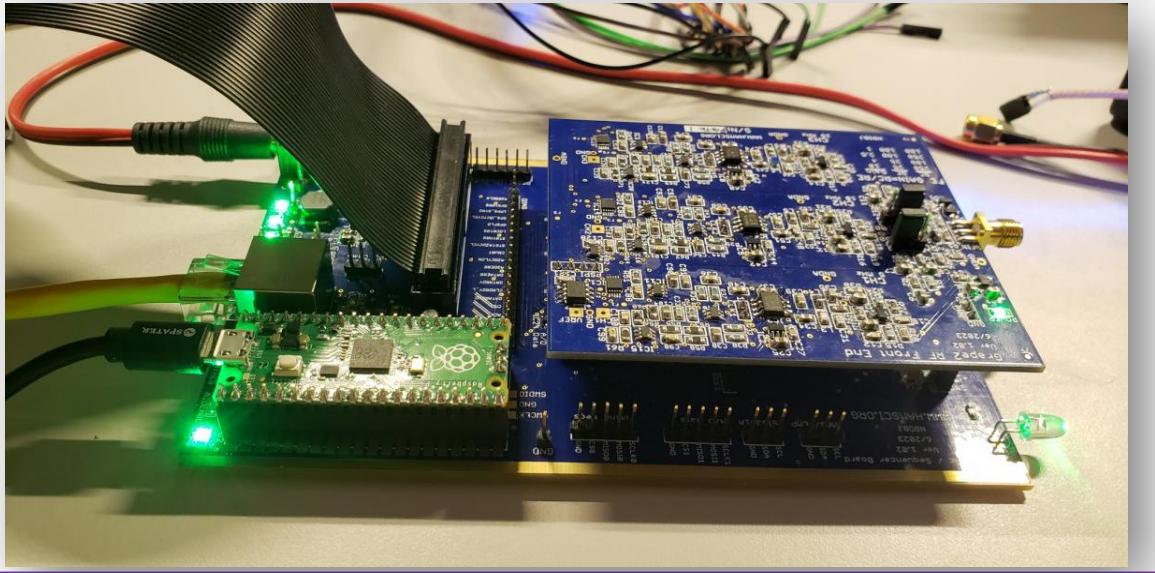
Outline

1. Background
2. G2Console
3. CSV2DRF
4. Grape 2 Spectrogram

cuong.nguyen@scranton.edu

Grape 2

- The HamSCI PSWS project is a distributed array of radio receivers.
- It records changes in the Earth's ionosphere by monitoring, on a 24/7 basis, frequency standard stations such as WWV/H and CHU.
- Grape 2 is a GPS-disciplined 3-channel receiver



THE UNIVERSITY OF SCRANTON
A JESUIT UNIVERSITY

cuong.nguyen@scranton.edu

Outline

1. Background
2. **G2Console**
3. CSV2DRF
4. Grape 2 Spectrogram

cuong.nguyen@scranton.edu

G2Console

- Lightweight terminal-based interface
- Use cases:
  - Display Real-time Data
  - Control Data Collection System

```
                    Grape2 Console v12.3
                      Node: N0001003
                     Firmware Versions
                        RasPi 0.7.13
                        Pico  0.5.16

GPS Date/Time          03/19/2024 08:38:00

UBLOX GPS          Fix       #Sats     PDOP
                   3D         19       1.09

RF Gain          Latitude       Longitude     Elevation(m)
  10             41.405375     -75.658169        269.8

                 Radio 1        Radio 2         Radio 3
Beacon            WWV5           WWV10           WWV15

Signal Level      Vpeak          Vpeak           Vpeak
MAX 1 hr        0.036170       0.521667        0.274811
Current         0.025724       0.272568        0.184304
MIN 1 hr        0.002082       0.025275        0.013205

Frequency           Hz             Hz              Hz
MAX 1 hr      5000000.659   10000000.660    15000000.533
Current       5000000.099   10000000.062    15000000.081
MIN 1 hr      4999998.955    9999999.845    14999999.813

Temp(C)         Local          Remote
                0.0            0.0

Magnetometer
B Field         X(uT)          Y(uT)           Z(uT)
MAX 1 hr         0.000          0.000           0.000
Current          0.000          0.000           0.000
MIN 1 hr         0.000          0.000           0.000

     <ctrl-x> = terminate Data Controller
     <ctrl-p> = toggle for 1Hr/24Hr Min/Max
```

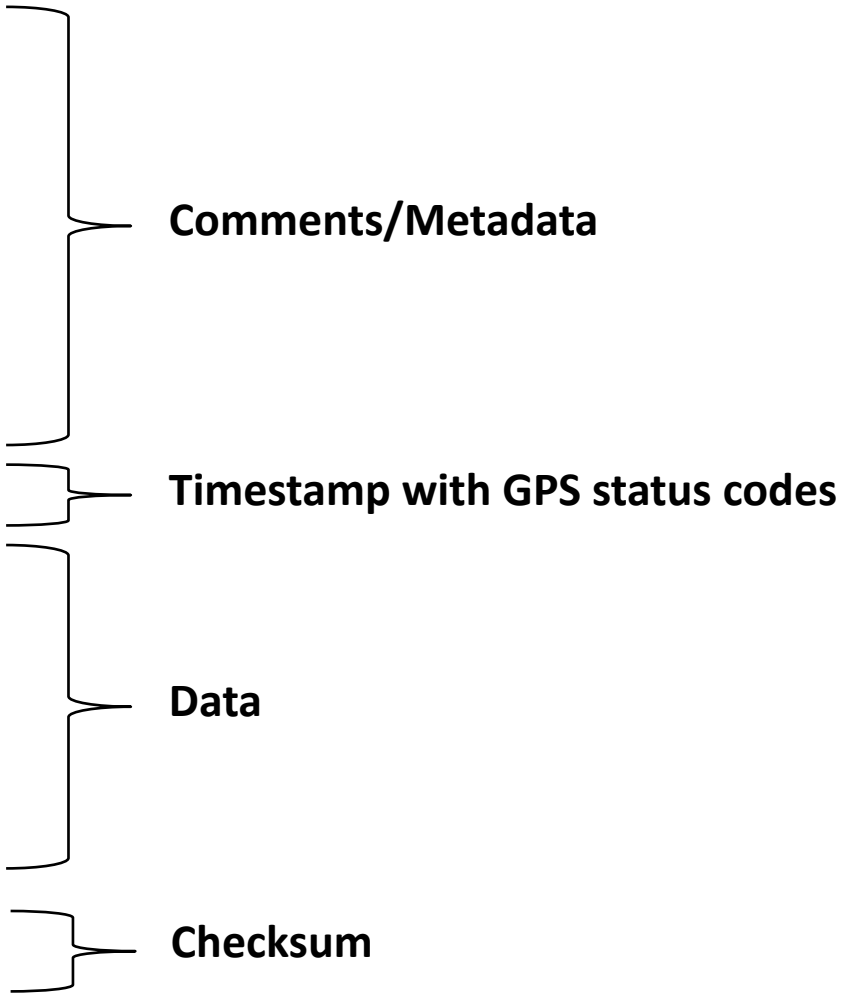
Outline

1. Background
2. G2Console
3. **CSV2DRF**
4. Grape 2 Spectrogram
5. TODOs

cuong.nguyen@scranton.edu

Grape 2 CSV Format

```
# RFGain                        10
# RadioID1                      G2R1
# <SNIP>
# A/D Sample Rate               8000
# A/D Zero Cal Data             7ee9,7f10,7f05
T20240408000000L3D2
7f32,7ef0,8103
7edc,8518,8425
7edc,8518,8425
<SNIP>
C08f6dce2V
```



CSV Files

- Benefits:
  - Simplicity
  - Human readable
  - Readable with simple text editors
- Downsides
  - Scalability
  - No time-series support
  - Difficulty accessing metadata

Grape2 Digital RF Format

- **Features**:
  - Efficient Data Storage: Leverages HDF5 for compression and indexing.
  - Metadata Support: Allows for storing auxiliary data with time series data.

```
cuong@OPQ100305P09:~$ du -sh ~/G2DATA/N0001001/Srawdata/
9.7G    /home/cuong/G2DATA/N0001001/Srawdata/
cuong@OPQ100305P09:~$ du -sh ~/G2DATA/N0001001/Sdrf/
8.1G    /home/cuong/G2DATA/N0001001/Sdrf/
```

```
OBS2024-04-08T00-00
└── ch0
    ├── 2024-04-08T00-00-00
    │   ├── rf@1712534400.000.h5
    │   ├── rf@1712534460.000.h5
    │   ├── rf@1712534520.000.h5
    │   ├── <SNIP>
    ├── 2024-04-08T01-00-00
    │   ├── rf@1712538000.000.h5
    │   ├── rf@1712538060.000.h5
    │   ├── rf@1712538120.000.h5
    │   ├── <SNIP>
    ├── 2024-04-08T02-00-00
    │   ├── rf@1712541600.000.h5
    │   ├── rf@1712541660.000.h5
    │   ├── rf@1712541720.000.h5
    │   ├── <SNIP>
    ├── <SNIP>
    ├── drf_properties.h5
    └── metadata
        ├── 2024-04-08T00-00-00
        │   └── metadata@1712534400.h5
        └── dmd_properties.h5

27 directories, 1443 files
```


cuong.nguyen@scranton.edu

Polars

- Data analysis and manipulation library (like Pandas)
- Written in Rust
- Multi-threaded
- Lazy mode

```
Duration: 0:28:25.855000
Total number of allocations: 10214
Total number of frames seen: 3769
Peak memory usage: 35.2 MiB
```

Resource usage **before** integrating Polars

```
Duration: 0:02:23.835000
Total number of allocations: 11457
Total number of frames seen: 4452
Peak memory usage: 5.6 GiB
```

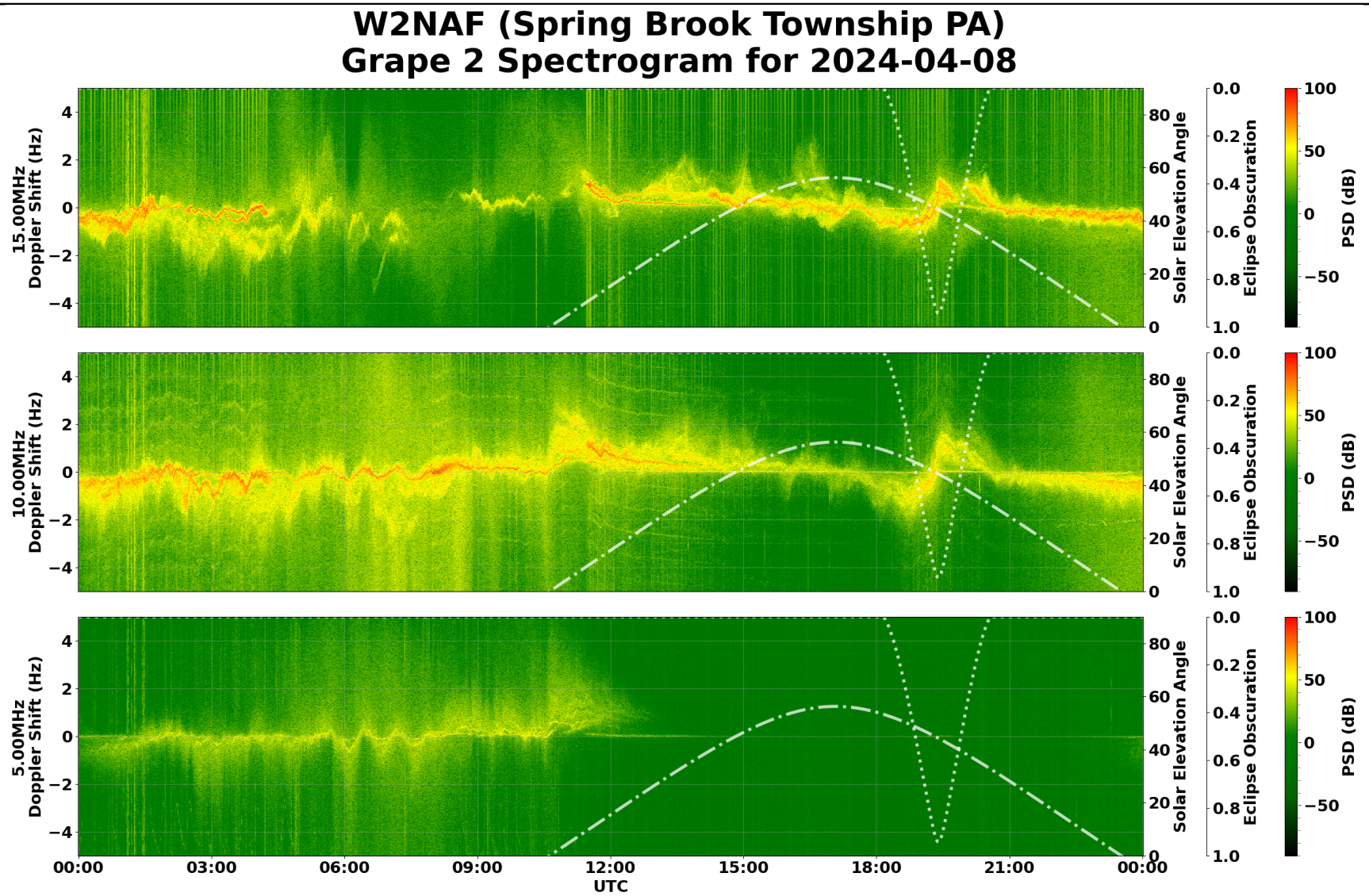
Resource usage **after** integrating Polars

Outline

1. Background
2. G2Console
3. CSV2DRF
4. **Grape 2 Spectrogram**
5. TODOs

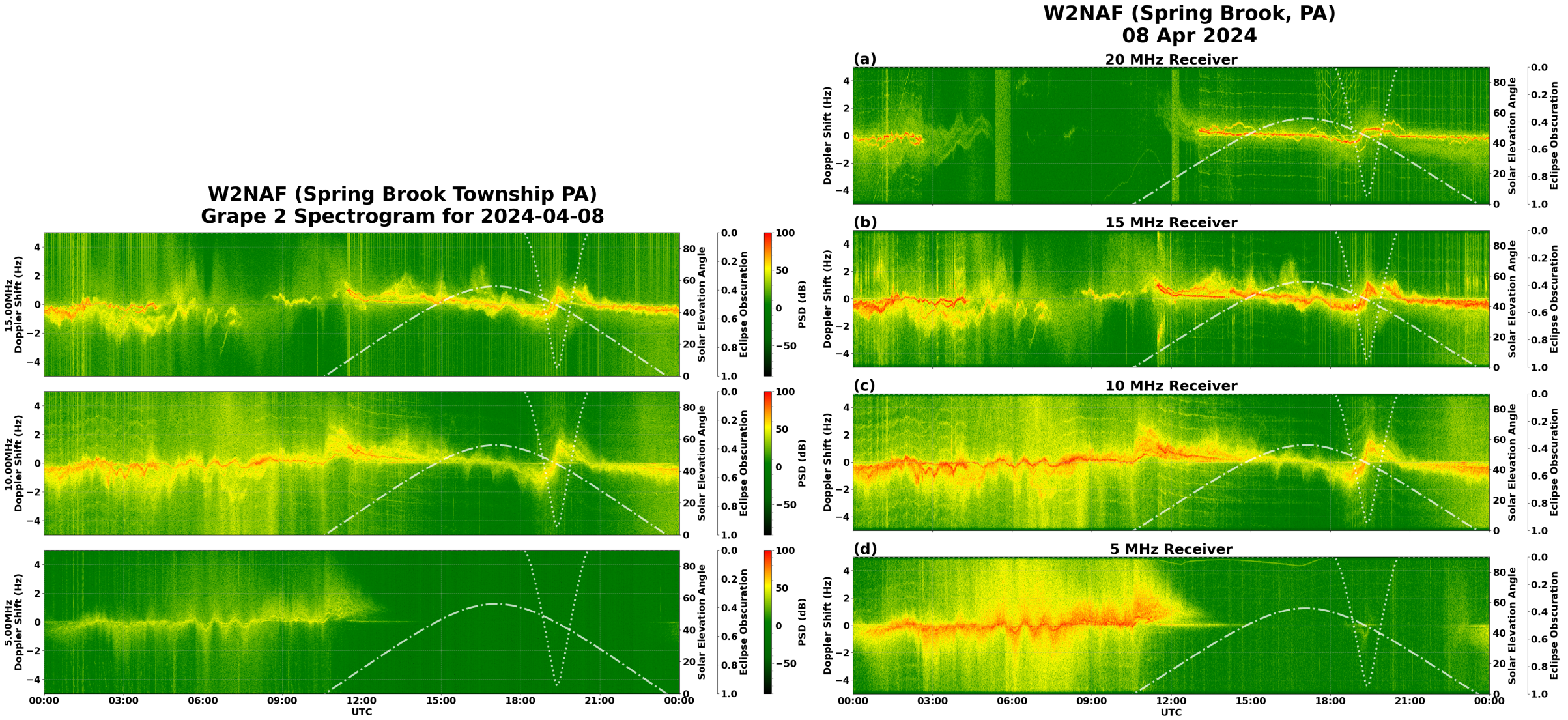
cuong.nguyen@scranton.edu

Grape 2 Spectrogram



cuong.nguyen@scranton.edu

Cross-Validation with WSPRDaemon



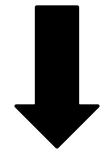
cuong.nguyen@scranton.edu

Resource Optimization

- Problems:
  - High computational time
  - VERY high memory usage (input is ~8GB uncompressed)
- Optimization techniques
  1. Batch processing: process in manageable chunks
  2. Resampling: resize dataset without causing distortion
  3. Caching: store resampled dataset on disk

```
Duration: 0:33:57.945000
Total number of allocations: 6275488
Total number of frames seen: 8825
Peak memory usage: 5.5 GiB
Python allocator: pymalloc
```

No cache and batch processing



```
Duration: 0:00:04.127000
Total number of allocations: 308477
Total number of frames seen: 8671
Peak memory usage: 275.0 MiB
Python allocator: pymalloc
```

With cache and batch processing

Acknowledgments

- We gratefully acknowledge the support of NSF Grants AGS-2230345, AGS-2045755, AGS-2002278, NASA Grant 80NSSC23K1322.
- The author acknowledges NIST, NRC, and their members for the operation of the WWV/WWVH/CHU stations.
- The author also acknowledges **John Gibbons, Bill Blackwell, Bill Engelke, Dr. Nathaniel Frissell, Dr. Mark Fenner, and the HamSCI community** for their support in the development of the system.
- Computations and plotting were done using open-source Python tools including but not up to SciPy, NumPy, Pandas, matplotlib, curses.

cuong.nguyen@scranton.edu

HamSCI Collaborative Efforts

The HamSCI Community is led by **The University of Scranton Department of Physics and Engineering W3USR**, in collaboration with **Case Western Reserve University W8EDU**, the **University of Alabama**, the **New Jersey Institute of Technology Center for Solar Terrestrial Physics K2MFF**, the **MIT Haystack Observatory**, **TAPR**, additional collaborating universities and institutions, and volunteer members of the **amateur radio** and citizen science communities. We are grateful for the financial support of the **United States National Science Foundation**, **NASA**, and **Amateur Radio Digital Communications (ARDC)**.

cuong.nguyen@scranton.edu

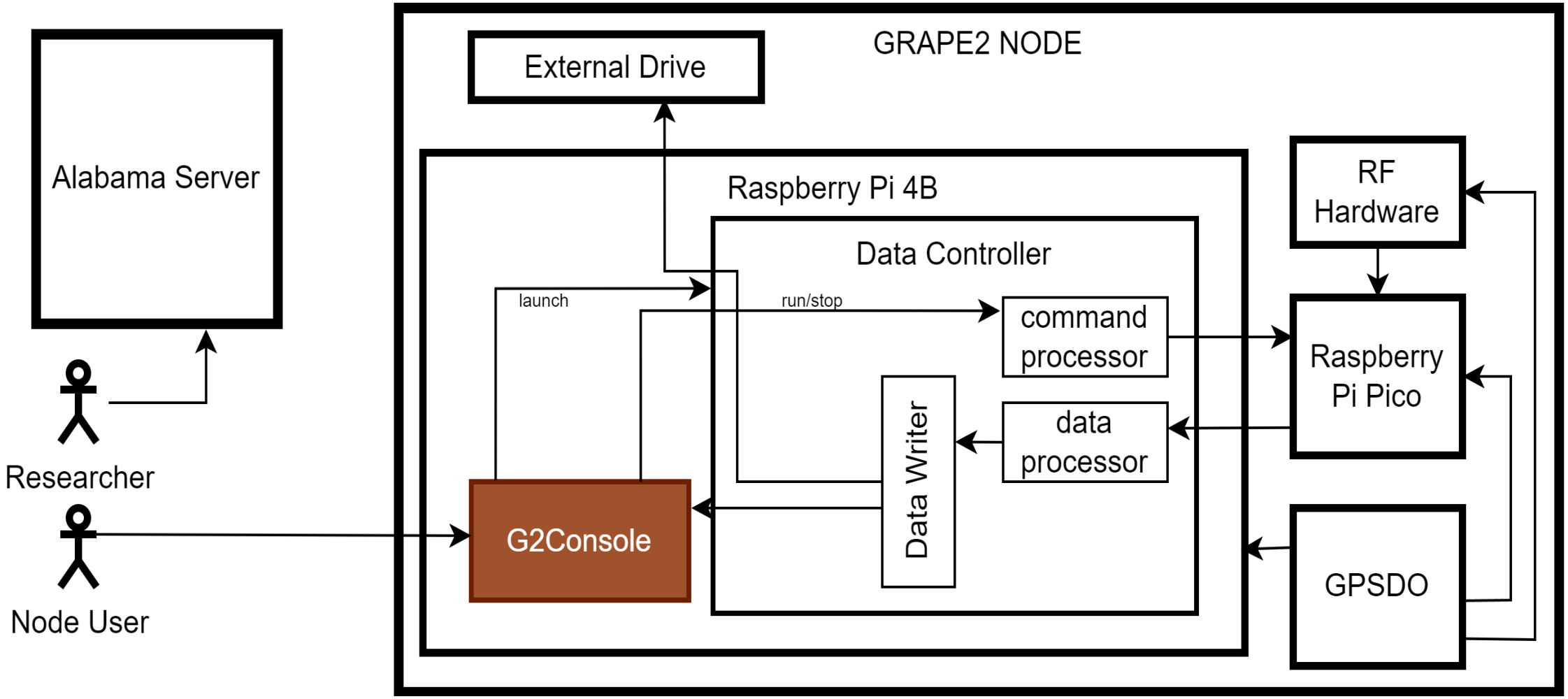
Thank you!

cuong.nguyen@scranton.edu

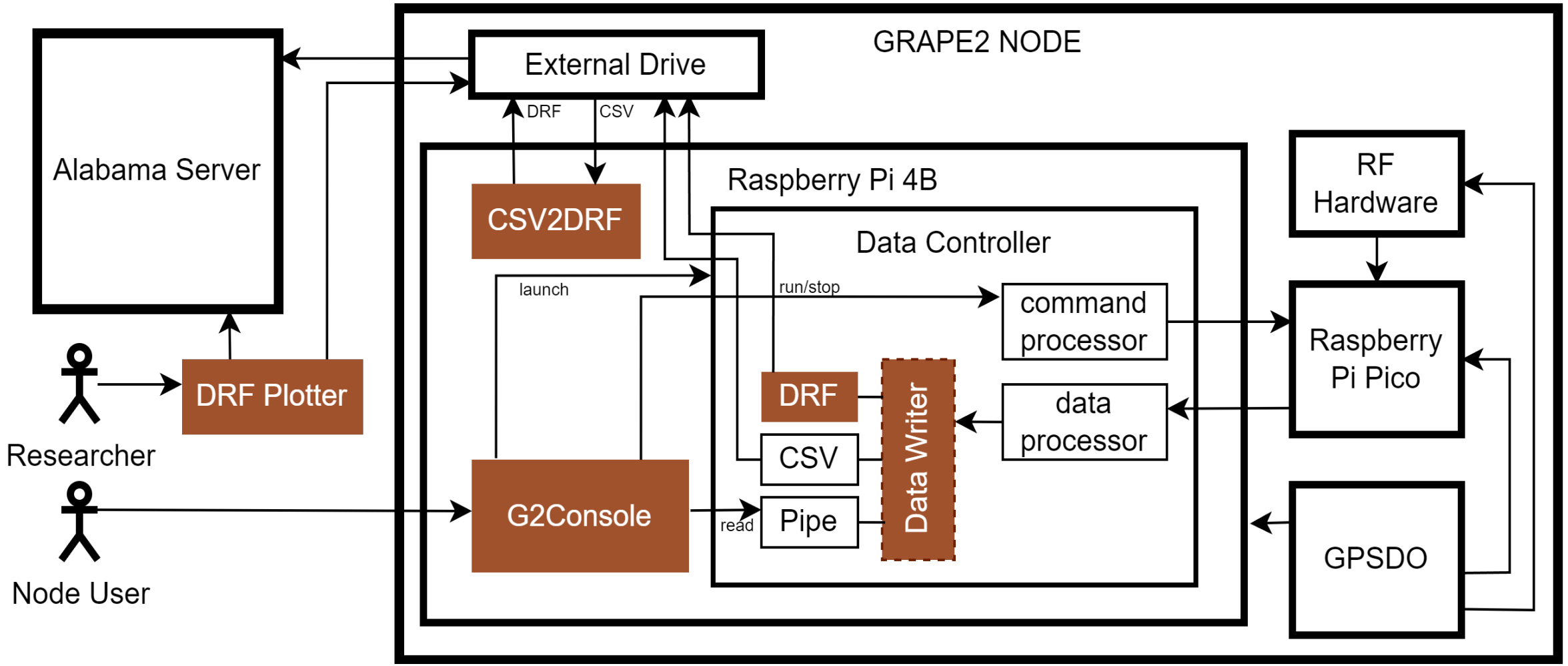
BACKUP SLIDES

cuong.nguyen@scranton.edu

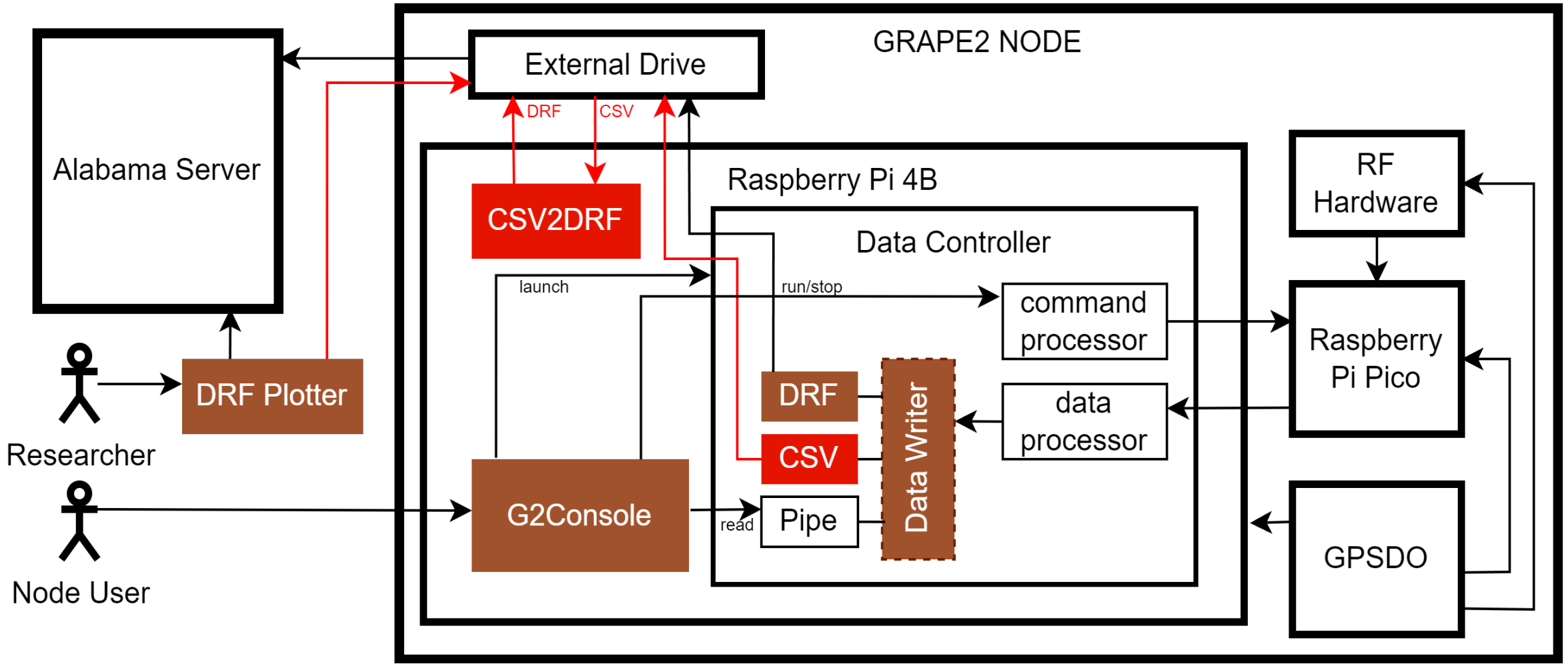
Grape2 System Architecture (Before Fall 2024)



Grape2 System Architecture (Now)



Grape2 System Architecture (Long-term)



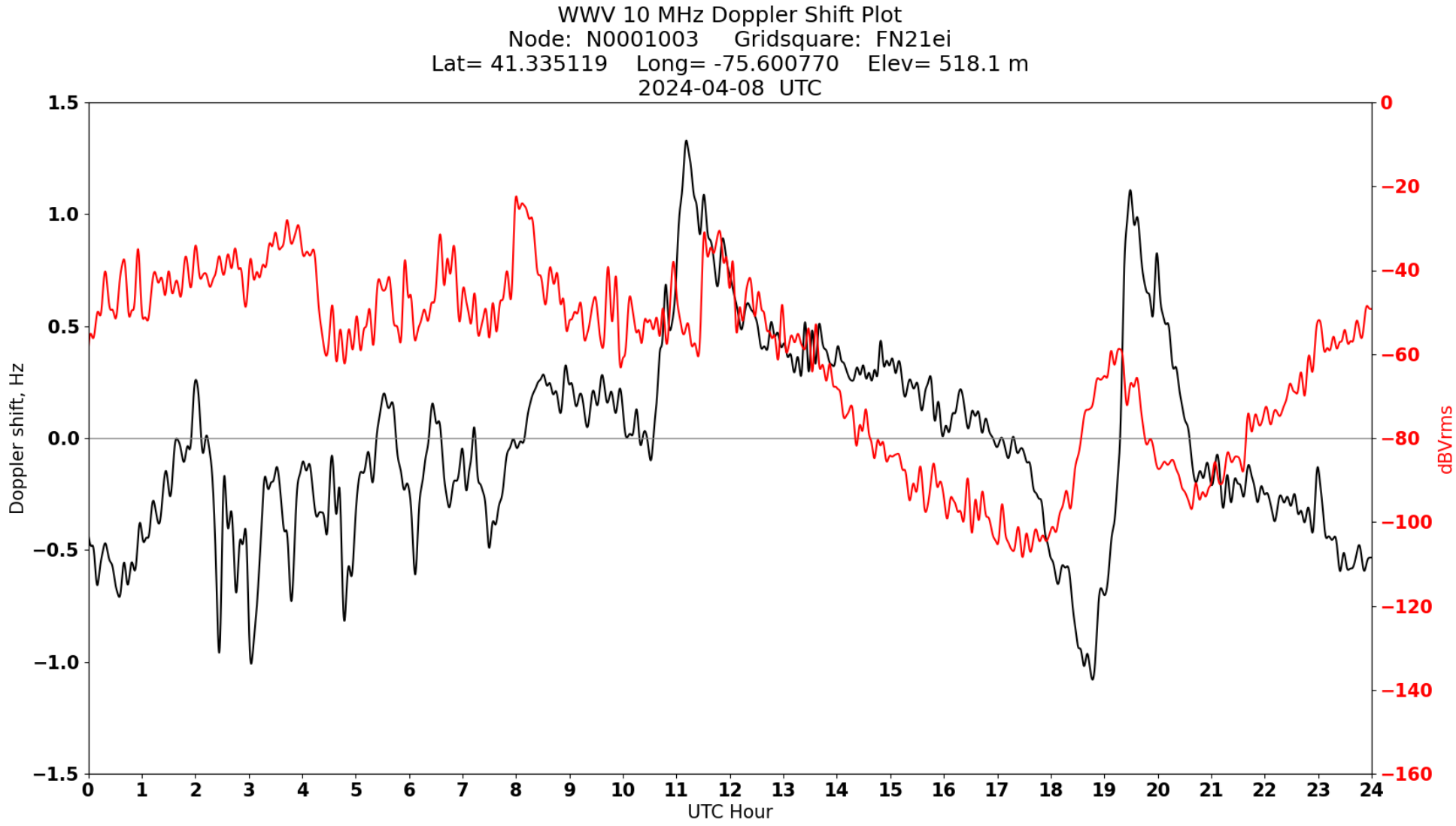
DIGITAL RF

- **What is DigitalRF?**
  - Open-source software library for time series data storage.
  - Designed for high-throughput scientific applications (e.g., radio astronomy, space weather).
  - Developed by MIT Haystack Observatory.

- **Features**:
  - Efficient Data Storage: Leverages HDF5 for compression and indexing.
  - Metadata Support: Allows for storing auxiliary data with time series data.
  - Extensibility: Easily integrates with other scientific computing tools.
- **Downside**: Specialized reader and writer functions

https://support.hdfgroup.org/documentation/

WWV → Spring Brook, PA on 10 MHz (Before Fall 2024)



Resampling and Caching

- Resize the dataset without distorting original signal (SciPy)
- Store the resampled data locally. Subsequent execution of the program will not re-read the original data (Pickle)
- **Benefits**:
  - Reduced input size
  - Configurability
  - Faster processing

Batch Processing

- **Objective:** process large datasets in manageable chunks
- **Benefits:**
  - Higher memory efficiency
  - Scalability
  - Configurability

```
Duration: 0:33:57.945000
Total number of allocations: 6275488
Total number of frames seen: 8825
Peak memory usage: 5.5 GiB
Python allocator: pymalloc
```

No cache, and batch processing

```
Duration: 0:00:04.127000
Total number of allocations: 308477
Total number of frames seen: 8671
Peak memory usage: 275.0 MiB
Python allocator: pymalloc
```

With cache, and batch processing

Memray

- Memray: a powerful memory profiler for Python, designed to track memory allocations and usage with detailed insights.

```
Duration: 0:12:19.298000
Total number of allocations: 6015037
Total number of frames seen: 8826
Peak memory usage: 18.1 GiB
Python allocator: pymalloc
```

No cache, no batch processing

curses

- What is it?
  - A library for creating terminal-based applications with advanced text-based UI.
  - Enables features like windows, menus, colors, and keyboard input handling.
- Why?
  - Lightweight
  - Efficient Input Handling
  - Minimal Dependencies

Outline

1. Background
2. G2Console
3. CSV2DRF
4. DRF Plotter
5. **TODOs**

cuong.nguyen@scranton.edu

TODOs

- Minor bug fixes -> CSV2DRF and Grape2Spectrogram
- Writing to DigitalRF directly, bypassing CSV
- Automatic data upload

Why DRF Over Other Database System?

- High-performance storage and retrieval
- Specialized metadata integration
- Optimized for scientific applications
- Why DRF over Big Data systems?
  - Specialization for time-series data
  - Ease of setup and use
  - Lower latency for real-time applications (e.g., data streaming)